**Amendments to the Claims**

Please amend Claims 1, 9, 15-18, 26, 32-35, and 49-54. The Claim Listing below will replace all prior versions of the claims in the application:

**Claim Listing**

1. (Currently Amended) A computer implemented scheduling method comprising the steps of:

   based on scheduling states, defining a set of static schedules for an application program, each static schedule including an assignment of tasks in the application program to processors;

   during run time, learning a cost of a set of static schedules based on performance of the application program; and

   designating a static schedule with a lowest cost as an optimal schedule for a scheduling state.

2. (Original) A scheduling method as claimed in Claim 1 wherein the cost of a set of static schedules is learned each time there is a change in scheduling state.

3. (Original) A scheduling method as claimed in Claim 1 wherein the cost of a set of static schedules is learned continuously during run time.

4. (Previously Presented) A scheduling method as claimed in Claim 1 further comprising:

   storing a set of all possible schedules associated with each schedule state; and

   upon a change of state, selecting the optimal schedule associated with the schedule state.

5. (Previously Presented) A scheduling method as claimed in Claim 4 wherein a selected schedule is the schedule with a lowest cost.

6. (Previously Presented) A scheduling method as claimed in Claim 4 wherein a selected schedule is the schedule with an unknown cost.

7. (Original) A scheduling method as claimed in Claim 6 wherein the schedule is randomly selected dependent on utility of exploration associated with the schedule.

8. (Original) A scheduling method as claimed in Claim 1 wherein the cost of a schedule is computed and stored after the schedule is executed.

9. (Currently Amended) A scheduling method as claimed in Claim 1 further comprising:
maintaining a task execution cost for each task in the application <u>program</u> for each scheduling state.

10. (Original) A scheduling method as claimed in Claim 9 wherein an optimal static schedule associated with a new scheduling state is computed using stored task execution costs.

11. (Previously Presented) A scheduling method as claimed in Claim 10 wherein the cost of an individual task is updated using stored task execution costs with recent schedule execution costs having more importance.

12. (Previously Presented) A scheduling method as claimed in Claim 10 wherein the cost of a schedule is updated using stored task execution costs with recent schedule execution costs having more importance.

13. (Previously Presented) A scheduling method as claimed in Claim 1 further comprising:
predicting the cost of a schedule dependent on stored task execution costs.

14. (Original) A scheduling method as claimed in Claim 13 wherein a schedule is selected for further exploration dependent on the predicted schedule cost.

15. (Currently Amended) A scheduling method as claimed in Claim 1 wherein the step of learning further comprises:

storing application program input data received during an active period in the application program; and

exploring optimal schedules while replaying stored input data during an idle period in the application program.

16. (Currently Amended) A scheduling method as claimed in Claim 15 wherein the step of learning further comprises:

concurrently executing a copy of an application program with identical input data on a processor other than another processor on which the application program is executing.

17. (Currently Amended) A scheduling method as claimed in Claim 16 wherein a change in optimized schedules is immediately reflected to a schedule analyzer for use in a next schedule change of the application program.

18. (Currently Amended) A scheduling system comprising:

a set of static schedules for an application program, the static schedules based on scheduling states, each static schedule including an assignment of tasks in the application program to processors; and

a schedule analyzer which:

during run time, learns a cost of the set of static schedules based on performance of the application program; and

designates a static schedule with a lowest cost as an optimal schedule for a scheduling state.

19. (Original) A scheduling system as claimed in Claim 18 wherein the schedule analyzer learns the cost of a set of static schedules each time there is a change in scheduling state.

20. (Original) A scheduling system as claimed in Claim 18 wherein the schedule analyzer learns the cost of a set of static schedules continuously during run time.

21. (Original) A scheduling system as claimed in Claim 18 further comprising:
    a list of schedule costs which stores an optimal schedule associated with each schedule state wherein upon a change of state the schedule analyzer selects the optimal schedule corresponding to the schedule state.

22. (Previously Presented) A scheduling system as claimed in Claim 21 wherein the schedule analyzer selects a schedule with a lowest cost.

23. (Original) A scheduling system as claimed in Claim 21 wherein the schedule analyzer selects a schedule with an unknown cost.

24. (Original) A scheduling system as claimed in Claim 23 wherein the schedule analyzer randomly selects a schedule dependent on utility of exploration associated with the schedule.

25. (Previously Presented) A scheduling system as claimed in Claim 18 wherein the schedule analyzer computes the cost of a schedule and stores a computed cost after the schedule is executed.

26. (Currently Amended) A scheduling system as claimed in Claim 18 further comprises:
    a task execution table which stores a task execution cost for each task in the application <u>program</u> for each scheduling state.

27. (Original) A scheduling system as claimed in Claim 26 wherein the schedule analyzer computes an optimal static schedule associated with a new scheduling state using stored task execution costs.

28. (Previously Presented) A scheduling system as claimed in Claim 27 wherein the schedule analyzer updates the cost of an individual task using a sliding window by discounting older execution results at an expense of more recent execution results.

29. (Previously Presented) A scheduling system as claimed in Claim 27 wherein the schedule analyzer updates the cost of a schedule using a sliding window by discounting older execution results at an expense of more recent execution results.

30. (Original) A scheduling system as claimed in Claim 18 wherein the schedule analyzer predicts the cost of a schedule dependent on stored task execution costs.

31. (Original) A scheduling system as claimed in Claim 30 wherein the scheduler analyzer selects a schedule for further exploration dependent on a predicted schedule cost.

32. (Currently Amended) A scheduling system as claimed in Claim 18 further comprising:

memory which stores application <u>program</u> input data received during an active period in the application <u>program</u>, stored application <u>program</u> input data allowing the schedule analyzer to explore optimal schedules while replaying the application <u>program</u> input data during an idle period in the application <u>program</u>.

33. (Currently Amended) A scheduling system as claimed in Claim 32 wherein the schedule analyzer provides a copy of an application <u>program</u> and the stored application <u>program</u> input data for concurrent execution on a processor other than another processor on which the application <u>program</u> is executing.

34. (Currently Amended) A scheduling system as claimed in Claim 33 wherein a change in optimized schedules is immediately reflected to the schedule analyzer for use in a next schedule change of the application program.

35. (Currently Amended) A scheduling system comprising:

a set of static schedules for an application program, the static schedules based on scheduling states, each static schedule including an assignment of tasks to processors;

means for learning which during run time, learns a cost of a set of static schedules based on performance of the application program; and

means for selecting which designates a static schedule with a lowest cost as an optimal schedule for a scheduling state.

36. (Original) A scheduling system as claimed in Claim 35 wherein the means for learning learns the cost of a set of static schedules is learned each time there is a change in scheduling state.

37. (Original) A scheduling system as claimed in Claim 35 wherein the means for learning learns the cost of a set of static schedules continuously during run time.

38. (Original) A scheduling system as claimed in Claim 35 further comprising:

a list of schedule costs which stores an optimal schedule associated with each schedule state wherein upon a change of state the means for analyzing selects the optimal schedule associated with the schedule state.

39. (Previously Presented) A scheduling system as claimed in Claim 38 wherein the means for selecting selects a schedule with a lowest cost.

40. (Original) A scheduling system as claimed in Claim 38 wherein the means for selecting selects a schedule with an unknown cost.

41. (Original) A scheduling system as claimed in Claim 40 wherein the means for selecting randomly selects a schedule dependent on utility of exploration associated with the schedule.

42. (Previously Presented) A scheduling system as claimed in Claim 35 wherein the means for selecting computes the cost of a schedule and stores a computed cost after the schedule is executed.

43. (Original) A scheduling system as claimed in Claim 35 further comprises:

    a task execution table which stores a task execution cost for each task in the application <u>program</u> for each scheduling state.

44. (Original) A scheduling system as claimed in Claim 43 wherein the means for selecting computes an optimal static schedule associated with a new scheduling state is using stored task execution costs.

45. (Previously Presented) A scheduling system as claimed in Claim 44 wherein the means for selecting updates the cost of an individual task using a sliding window by discounting older execution results at a expense of more recent execution results.

46. (Previously Presented) A scheduling system as claimed in Claim 44 wherein the means for selecting updates the cost of a schedule using a sliding window by discounting older execution results at a expense of more recent execution results.

47. (Original) A scheduling system as claimed in Claim 35 wherein the means for selecting predicts the cost of a schedule dependent on stored task execution costs.

48. (Previously Presented) A scheduling system as claimed in Claim 47 wherein the means for selecting selects a schedule for further exploration dependent on a predicted cost for the schedule.

49. (Currently Amended) A scheduling system as claimed in Claim 35 wherein the further comprising:

memory which stores application program input data received during an active period in the application program, the application program input data allowing the scheduling analyzer to explore optimal schedules while replaying the application program input data during an idle period in the application program.

50. (Currently Amended) A scheduling system as claimed in Claim 49 wherein an on-line scheduling system provides a copy of an application program and the application program input data for concurrent execution on a processor other than another processor on which the application program is executing.

51. (Currently Amended) A scheduling system as claimed in Claim ~~18~~ 50 wherein a change in ~~a~~ optimized schedules is immediately reflected to the means for analyzing for use in a next schedule change of the application program.

52. (Currently Amended) A computer system comprising:

a central processing unit connected to a memory system by a system bus;

an I/O system, connected to the system bus by a bus interface; and

a scheduling system routine located in the memory system which:

based on scheduling states, defines a set of static schedules for an application program, each static schedule including an assignment of tasks in the application program to processors;

during run time, learns a cost of a set of static schedules based on performance of the application program; and

designates a static schedule with a lowest cost as an optimal schedule for a scheduling state.

53. (Currently Amended) A computer program product for system scheduling, the computer program product comprising a computer usable medium having computer readable program code thereon, including program code which:

based on scheduling states, defines a set of static schedules for an application program, each static schedule including an assignment of tasks in the application program to processors;

during run time, learns a cost of a set of static schedules based on performance of the application program; and

designates a static schedule with a lowest cost as an optimal schedule for a scheduling state.

54. (Currently Amended) The method of claim 1, wherein the performance of the application program is based on time to complete one iteration of the application program.